

Semantic Web Technologies I (WS13/14): Lightweight Semantics for Web Services and Web APIs

Maria Maleshkova

<http://semantic-web-grundlagen.de>

Institut AIFB, KIT



Lecture Outline

1. Web Services

1. Background
2. SOAP
3. WSDL

2. Web APIs

3. Lightweight Semantics

Introduction to Web Services

■ Definition of IBM:

- „Web services are a new breed of Web application. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes.”

Introduction to Web Services

■ Definition of W3C:

- „A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Webrelated standards.“

Simple Scenario



History

- Web services evolved from previous technologies that served the same purpose such as RPC, ORPC (DCOM, CORBA and JAVA RMI)
- Web Services were intended to solve two main problems:
 - Interoperability
 - Complexity

Interoperation of Platforms

- Software inter-operation:
 - Across the **Internet** and between **dissimilar platforms** (hardware platforms, operating systems)
- The **UNIX** family of operating systems introduced 'remote procedure call' (**RPC**)
- An inter-system RPC included in plans for **OSF** Distributed Computing Environment (**DCE**)
- **Microsoft**, for Windows, reused DCE/RPC in the object-oriented Common Object Model (**COM**)
- The **OMG** (Object Management Group) worked on the Common Object Request Broker Architecture (**CORBA**)

Interoperation Between Companies

- Collaboration across corporations was an issue because distributed systems such as CORBA and DCOM used non-standard ports
- Web Services use HTTP as a transport protocol and most of the firewalls allow access through port 80 (HTTP), leading to easier and dynamic collaboration

Complexity

- Web Services is a developer-friendly service system
- Most of the above-mentioned technologies such as RMI, COM, and CORBA involve a whole learning curve
- New technologies and languages have to be learnt to implement these services

Web Services and Business Services

- Web Services seek to use Web technology (HTTP, URIs, XML) to
 - Represent and make available business services:
 - B2B (Business-to-Business)
 - B2C (Business-to-Customer)
 - Etc.

Business-to-Business Services

- XML was gaining ground in development of EDI (Electronic Data Interchange):
 - UN/EDIFACT was United Nations (later ISO) pre-XML standard structuring data for “Administration, Commerce and Transport”
 - ebXML subsequent joint initiative between UN & OASIS (Organization for the Advancement of Structured Information Standards) for Electronic Business using XML

Business-to-Customer Services

- Early Web offered limited but compelling means to offer customer services, e.g. **Amazon (since 1994)**
- HTTP and **HTML support** (e.g. forms) for browser, but not clear for other applications



Web Service Definition (revised)

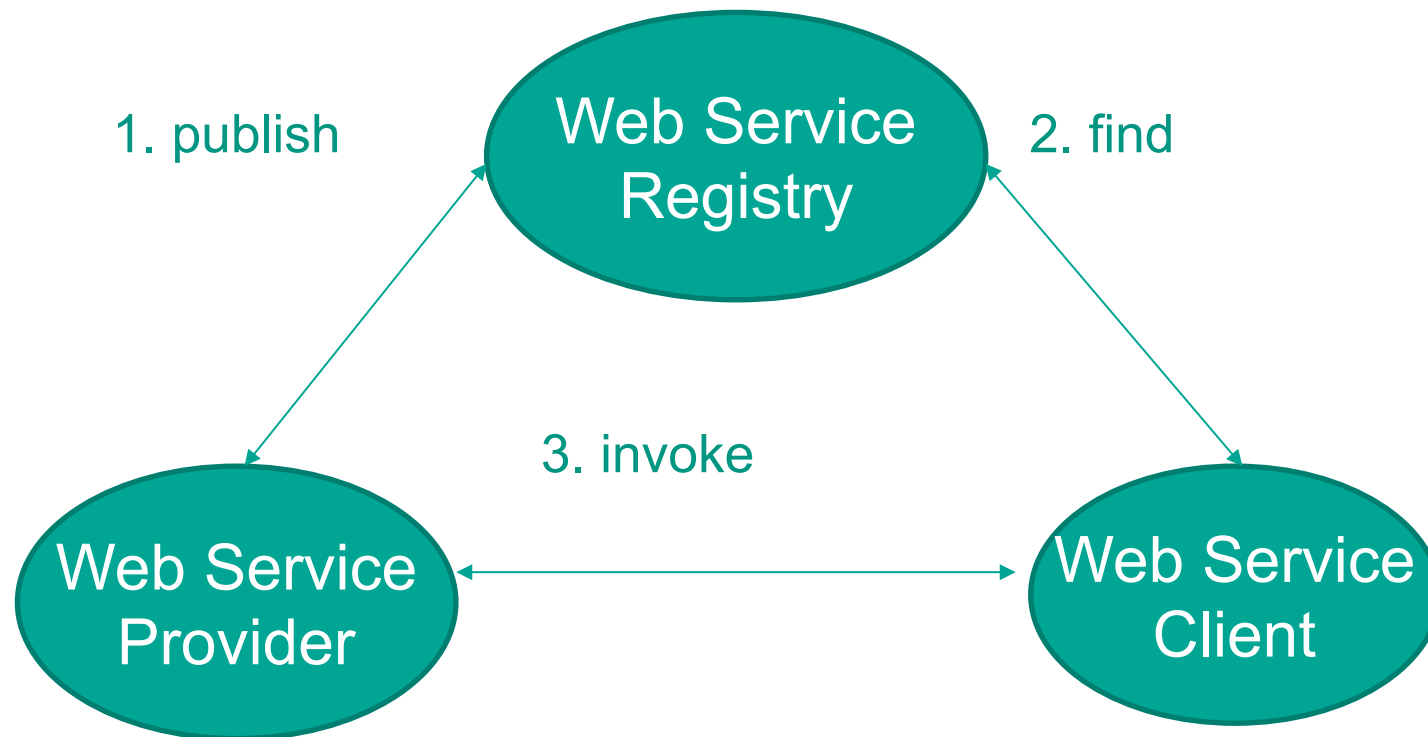
- An application component that:
 - Communicates via open protocols (HTTP, SMTP, etc.)
 - Processes XML messages framed using SOAP
 - Describes its messages using XML Schema
 - Provides an endpoint description using WSDL
 - Can be discovered using UDDI

The Web Service Model

- The Web Services architecture is based upon the interactions between three roles:
 - Service provider
 - Service registry
 - Service requestor
- The interactions involve the:
 - Publish operations
 - Find operation
 - Invoke operations

The Web Service Model

The Web Services model follows the *publish*, *find*, and *invoke* paradigm.



Web Services Components

- XML – eXtensible Markup Language – A uniform data representation and exchange mechanism.
- SOAP – Simple Object Access Protocol – A standard way for communication.
- UDDI – Universal Description, Discovery and Integration specification – A mechanism to register and locate WS based application.
- WSDL – Web Services Description Language – A standard meta language to described the services offered.

Example – A Simple Web Service

- A buyer (which might be a simple client) is ordering goods from a seller service.
- The buyer finds the seller service by searching the UDDI directory.
- The seller service is a Web Service whose interface is defined using Web Services Description Language (WSDL).
- The buyer is invoking the order method on the seller service using Simple Object Access Protocol (SOAP) and the WSDL definition for the seller service.
- The buyer knows what to expect in the SOAP reply message because this is defined in the WSDL definition for the seller service.

Lecture Outline

1. Web Services

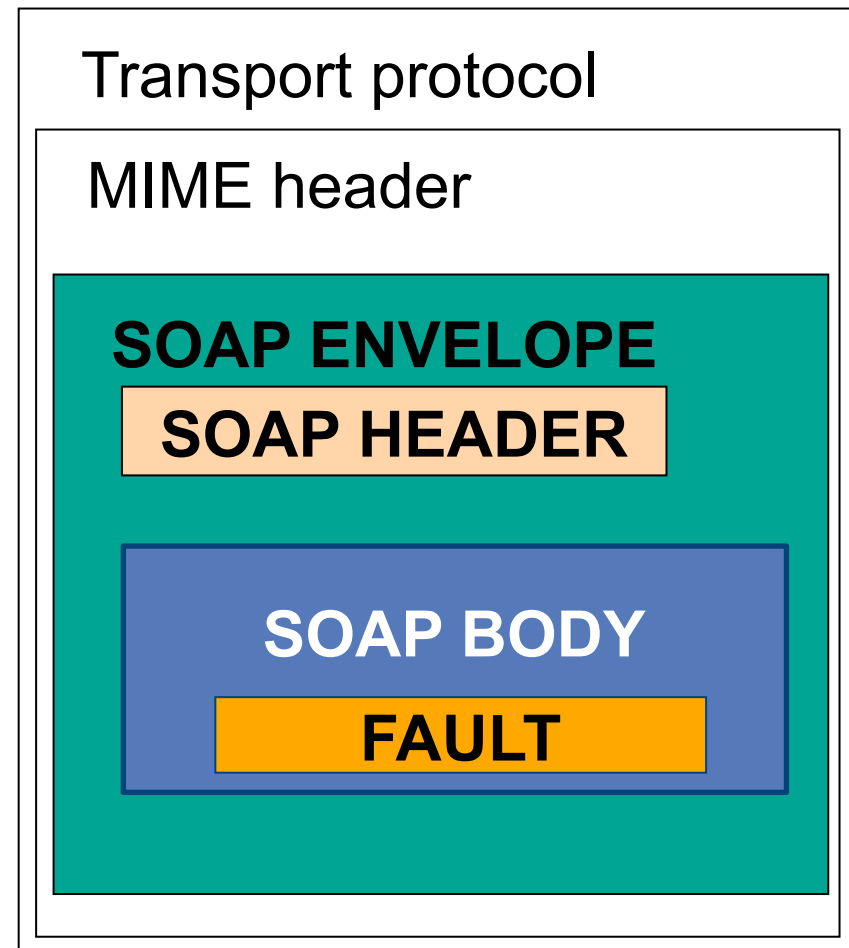
1. Background
2. SOAP
3. WSDL

2. Web APIs

3. Lightweight Semantics

SOAP

- Originally the Simple Object Access Protocol (now just SOAP) defines:
 - A messaging format, using XML, consisting of an envelope, composed of
 - Header,
 - Body,
 - Set of standard elements and attributes;
 - A processing model, agnostic of transport method, but practically mainly used over HTTP



SOAP Messaging Format

POST /InStock HTTP/1.1

Host: www.example.org

Content-Type: application/soap+xml; charset=utf-8 Content-Length: 150

```
<?xml version='1.0' ?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
    ...
  </soap:Header>
  <soap:Body>
    ...
  </soap:Body>
</soap:Envelope>
```

SOAP Request Example

```
<?xml version='1.0' ?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
    ...
  </soap:Header>
  <soap:Body>
    ...
  </soap:Body>
</soap:Envelope>
```

←

```
<stock:GetStockPriceRequest
  xmlns:stock=
    "http://www.example.org/stock">
  <stock:StockName>IBM</stock:StockName>
</stock:GetStockPriceRequest>
```

Note that 'stock' is a
schema defined by the
service provider

SOAP Response Example

```
<?xml version='1.0' ?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
    ...
  </soap:Header>
  <soap:Body>
    ...
  </soap:Body>
</soap:Envelope>
```

←

```
<stock:GetStockPriceResponse
  xmlns:stock=
    "http://www.example.org/stock">
  <stock:Price>34.5</stock:Price>
</stock:GetStockPriceResponse>
```

Again, within the schema
defined by the service
provider

SOAP Fault Format

```
<?xml version='1.0' ?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
    ...
  </soap:Header>
  <soap:Body>
    ...
  </soap:Body>
</soap:Envelope>
```

```
<env:Fault>
  <env:Code>
    <env:Value>...</env:Value>
  </env:Code>
  <env:Reason>
    <env:Text>...</env:Text>
  </env:Reason>
  <env:Detail> ...
  </env:Detail>
</env:Fault>
```

The Fault structure is part of the SOAP schema

Lecture Outline

1. Web Services

1. Background
2. SOAP
3. WSDL

2. Web APIs

3. Lightweight Semantics

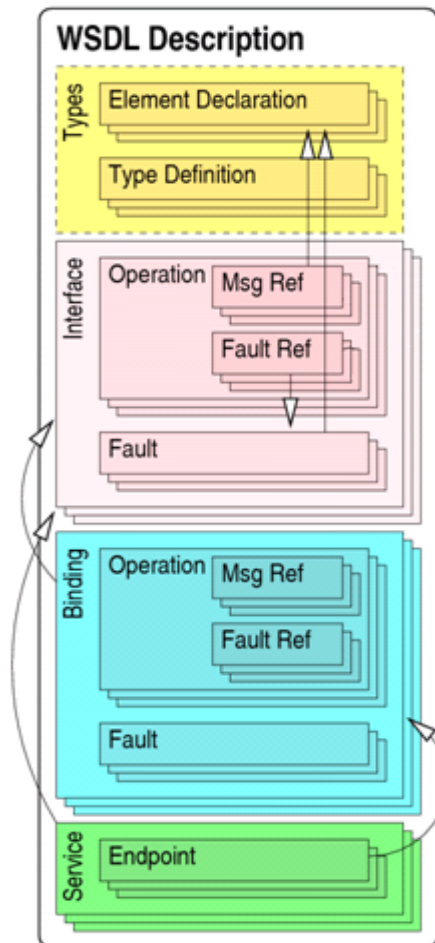
WSDL

- WSDL stands for Web Services Description Language
- WSDL is an XML vocabulary for describing Web services. It allows developers to describe Web Services and their capabilities, in a standard manner
- WSDL specifies what a request message must contain and what the response message will look like in unambiguous notation. In other words, it is a contract between the XML Web service and the client who wishes to use this service
- In addition to describing message contents, WSDL defines where the service is available and what communications protocol is used to talk to the service

WSDL

- Web Services Description Language (WSDL) developed in 2000 by **Microsoft** and **IBM** to:
 - “[describe] network services as a set of **endpoints** operating on messages [...]”
 - The **operations** and **messages** are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (**services**)”

WSDL Overview



- **types** describes the messages that the service will send and receive
(XML Schema Definitions, XSD, are used for definitions)
- **interfaces** describes *what* abstract functionality the service provides;
(collection of operations, linked to messages & faults, formerly called PortTypes)
- **bindings** describe *how* to access the service, i.e., which interfaces are supported by a given service
- **services** describe *where* to access
(endpoint = URL)

WSDL Example

```
<?xml version="1.0"?>
<definitions name=...
    targetNamespace=...
    xmlns="http://www.w3.org/ns/wsdl">

    <types>
        <schema targetNamespace=...
            xmlns:xs="http://www.w3.org/2000/10/XMLSchema">
            <element name=... type=...>
            <complexType name=...>
            </complexType>
            ...
        </schema>
    </types>
    ...
</definitions>
```

WSDL Example

```
<?xml version="1.0"?>
<definitions name=...
    targetNamespace=...
    xmlns="http://www.w3.org/ns/wsdl">

    <types> ... </types>

    <interface>
        <fault name=... element=... />
        <operation name=...>
            <pattern="http://www.w3.org/ns/wsdl/in-out">
                ...
            <input messageLabel=... element=... />
            <output messageLabel=... element=... />
        </operation>
    </interface>
```

WSDL Example

```
<?xml version="1.0"?>
<definitions name=...
    targetNamespace=...
    xmlns="http://www.w3.org/ns/wsdl">

    <types> ... </types>

    <interface> ... </interface>

    <binding name=...
        interface=...
        type="http://www.w3.org/ns/wsdl/soap"
        xmlns:wsoap=
            "http://www.w3.org/ns/wsdl/soap"
        wsoap:protocol=
            "http://www.w3.org/2003/05/soap/bindings/HTTP/">
    </binding>
```

WSDL Example

```
<?xml version="1.0"?>  
<definitions name=...  
    targetNamespace=...  
    xmlns="http://www.w3.org/ns/wsd1">
```

```
    <types> ... </types>
```

```
    <interface> ... </interface>
```

```
    <binding ...> ... </binding>
```

```
        <service name=...>  
            <endpoint name=...  
                binding=...  
                address=...  
            </service>
```

Lecture Outline

1. Web Services

1. Background
2. SOAP
3. WSDL

2. Web APIs

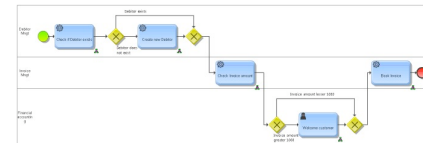
3. Lightweight Semantics

Web Services on the Web

- The Web currently contains 30 billion Web pages
- Nearly 100M active sites
 - 10 million new pages added each day
- The Web contains only 28,000 WSDL Web services (Seekda.com)
 - Verizon have around 4,000

Web Services on the Web

- “Traditional” Web services (WSDL, SOAP, WS-*)
 - Enable the publishing and consuming of functionalities of existing applications



- Web APIs
 - Enable the access to collections of resources



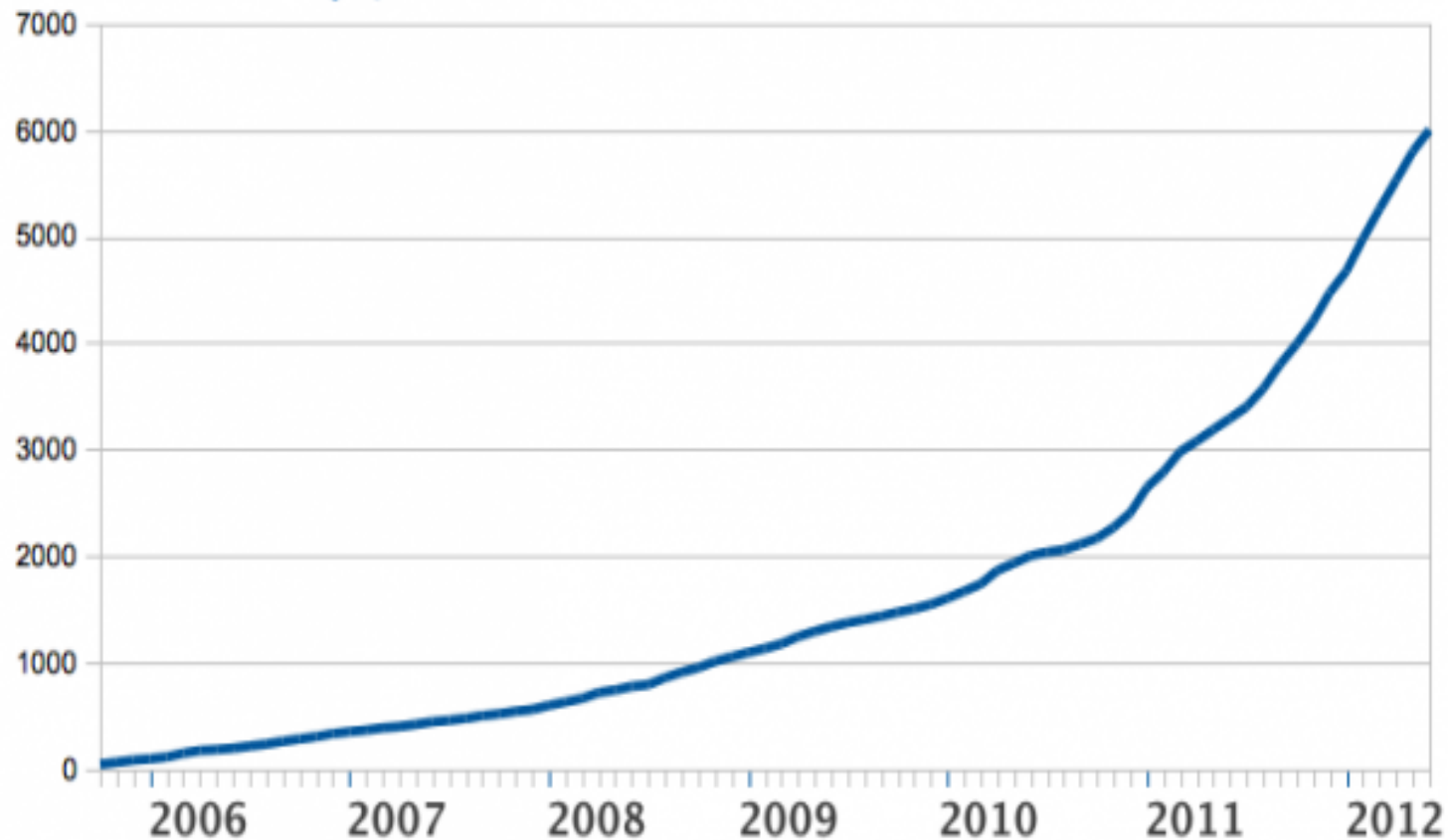
Why Web APIs?

9449
APIs

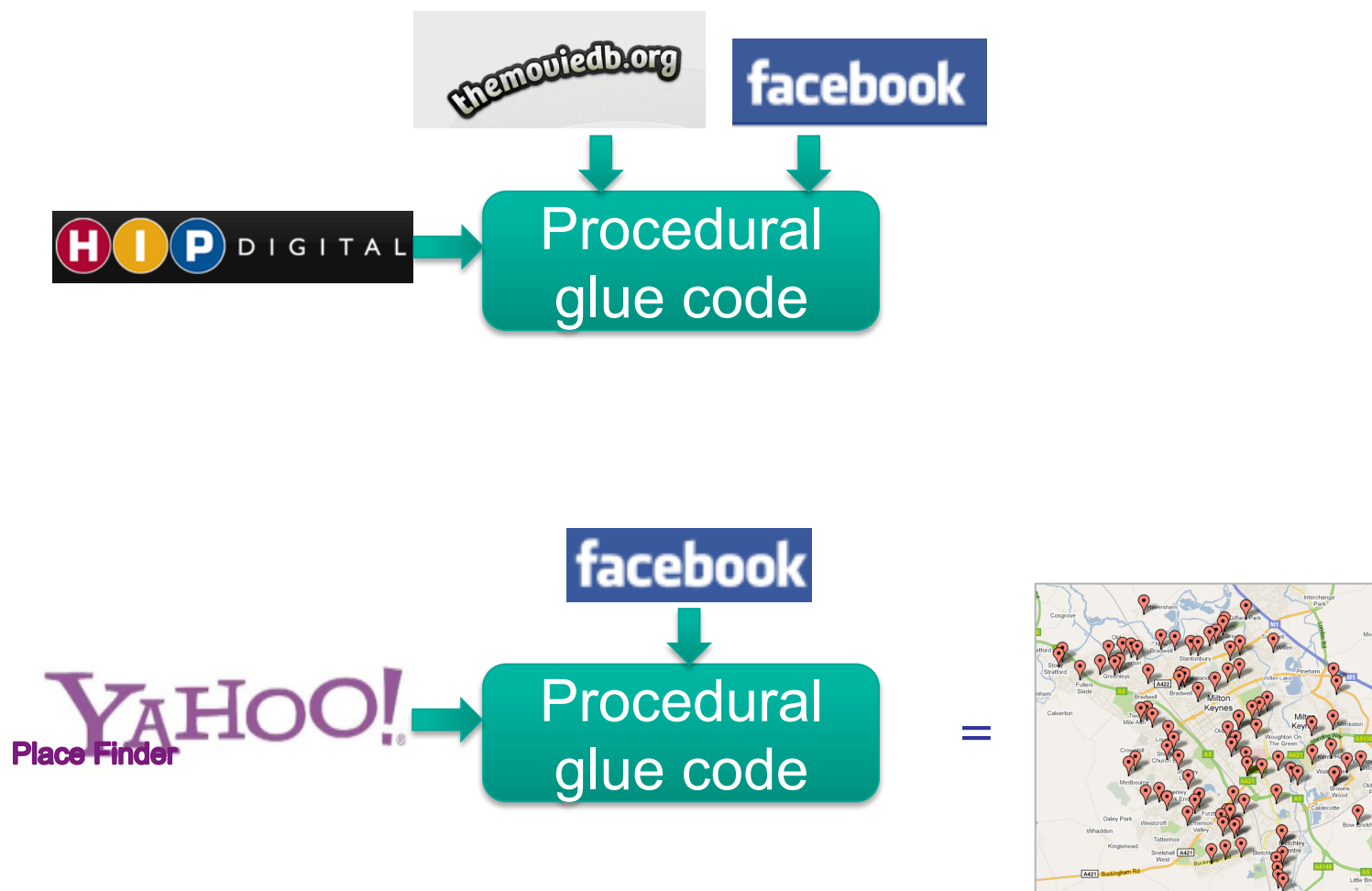
7107
Mashups



API Growth 2005 - 2012



Mashups



Why Web APIs?



API	Description	Category	Mashups
Google Maps	Mapping services	Mapping	2470
Twitter	Microblogging service	Social	781
YouTube	Video sharing and search	Video	671
Flickr	Photo sharing service	Photos	621
Amazon Product Advertising	Online retailer	Shopping	421
Facebook	Social networking service	Social	401
Twilio	Telephony service	Telephony	354
Twitter	Online radio service	Music	234
eBay	eBay Search service	Search	223
Google Search	Search services	Search	187
Microsoft Bing Maps	Mapping services	Mapping	175
Twilio SMS	SMS messaging service	Messaging	173
del.icio.us	Social bookmarking	Bookmarks	161
YAHOO! LOCAL Search	Search services	Search	145
Yahoo Maps	Mapping services	Mapping	136
Google Ajax Search	Web search components	Search	134
Google App Engine	Developer platform	Tools	123
411Sync	SMS, WAP, and email messaging	Messaging	120
foursquare	Social networking and city exploration	Social	102



API Doc

[Overview](#)
[User Authentication](#)
[Submissions \(Scrobbling\)](#)
[Radio API](#)
[Playlists](#)
[Downloads](#)
[REST requests](#)
[XML-RPC requests](#)

API Methods

Album

[album.addTags](#)
[album.getInfo](#)
[album.getTags](#)
[album.removeTag](#)
[album.search](#)

Artist

[artist.addTags](#)
[artist.getEvents](#)
[artist.getImage](#)
[artist.getInfo](#)
[artist.getPastEvents](#)
[artist.getPodcast](#)
[artist.getShouts](#)



Last.fm Web Services

[API](#) | [Feeds](#) | [Your API Account](#)

album.getInfo

Get the metadata for an album on Last.fm using the album name or a musicbrainz id. See [playlist.fetch](#) on how to get the album playlist.

e.g. http://ws.audioscrobbler.com/2.0/?method=album.getInfo&api_key=b25b959554ed76058ac220b7b2e0a026...

Params

artist (Optional) : The artist name in question
album (Optional) : The album name in question
mbid (Optional) : The musicbrainz id for the album
username (Optional) : The username for the context of the request. If supplied, the user's playcount for this album is included in the response.
lang (Optional) : The language to return the biography in, expressed as an ISO 639 alpha-2 code.
api_key (Required) : A Last.fm API key.

Auth

This service does **not** require authentication.

Sample Response

Web APIs Technologies

- Web APIs are based on a light technology stack \cong URIs, HTTP, XML/JSON
- Very much aligned with Web technologies
- Some are based on REST principles
 - Resource identification through URIs
 - Uniform interface
 - Self-descriptive messages
 - Stateful interactions through hyperlinks

Challenges with Web APIs

- There is not a widely used IDL
 - Locating services is hard
 - Their use requires human interpretation of semi-structured descriptions
- The semantics of the services are not described in a machine processable manner
- Prevents automating discovery, invocation, and composition of Web APIs

Lecture Outline

1. Web Services

1. Background
2. SOAP
3. WSDL

2. Web APIs

3. Lightweight Semantics

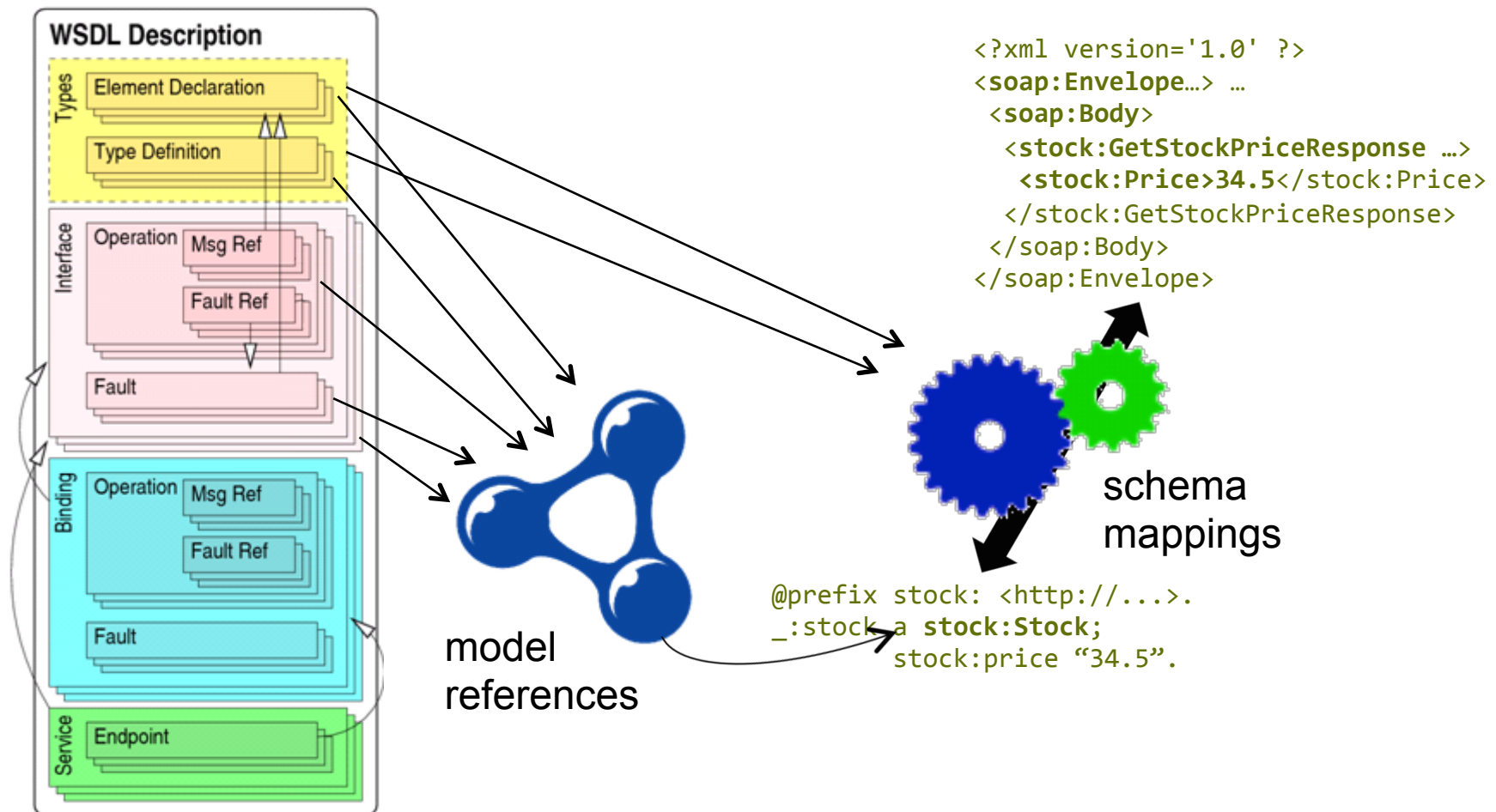
SAWSDL

- Semantic Annotations for WSDL and XML Schema (SAWSDL):
 - was developed by IBM with the semantics community
 - “defines how to add **semantic annotations** to various parts of a WSDL document such as
 - input and output **message** structures,
 - **interfaces**, and
 - **operations**”
 - became a W3C recommendation in 2007

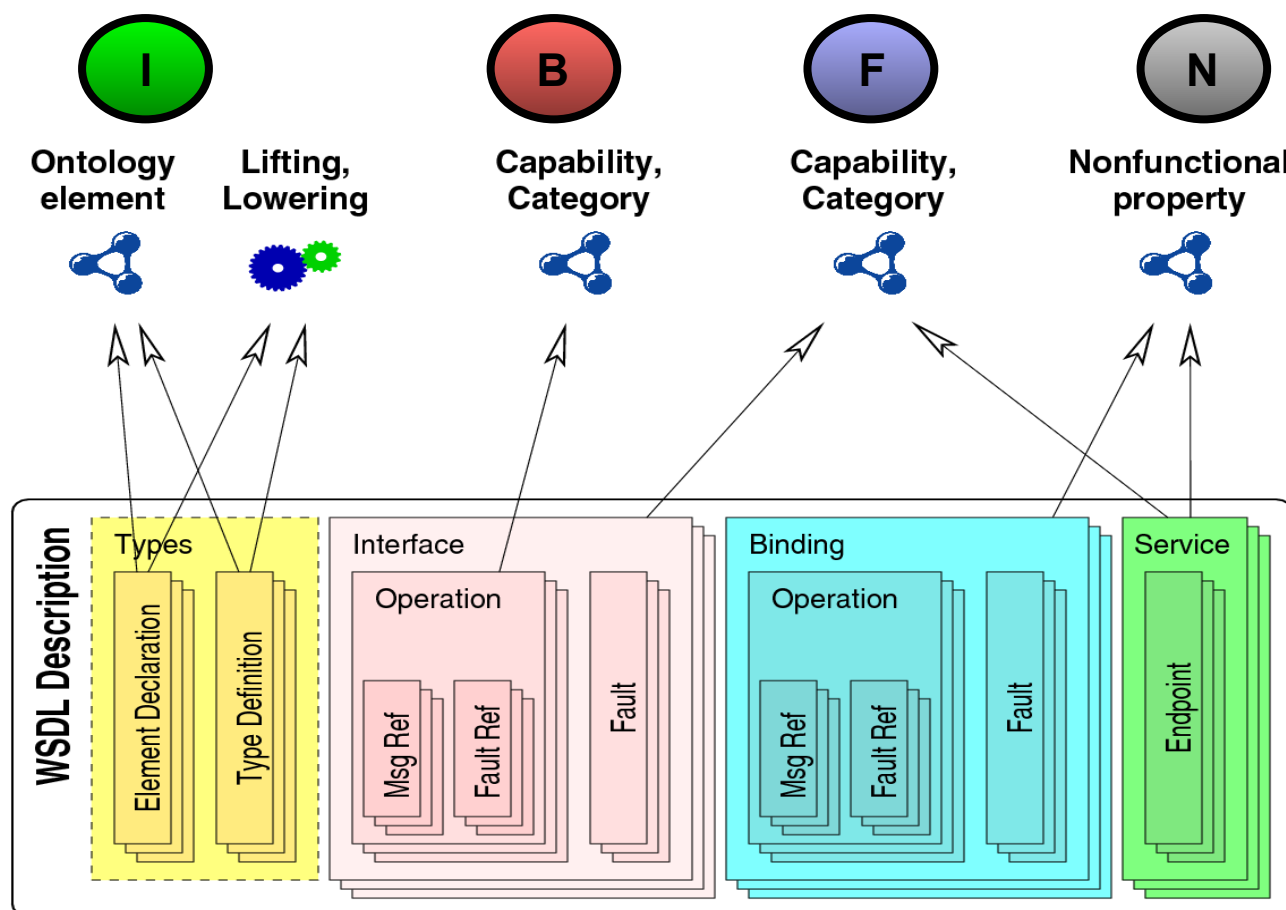
SAWSDL Overview

- SAWSDL essentially adds 3 attributes to WSDL:
 - **modelReference**: points to some external (i.e., non-XML, potentially RDF-based) semantic definition of the element
 - **liftingSchemaMapping**: defines how to convert from XML representations (e.g., of messages) to semantic form
 - **loweringSchemaMapping**: defines how to convert from semantic form to XML

SAWSDL Overview Diagram

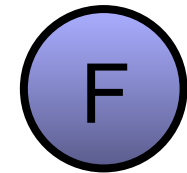


Kinds of Service Semantics



Functional Semantics

- For service discovery, composition
- Category
 - Functionality categorization
 - E.g. eCl@ss
 - Or tagging, folksonomies
- Capability
 - Precondition, Effect
 - Needs using some rule language (WSML, RIF, etc)



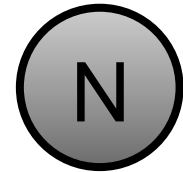
```
<wsdl:interface  
  name="BookStoreInterface"  
  sawsdl:modelReference=  
    "http://example.com/services/ecommerce/  
    books">  
  ...  
</wsdl:interface>
```

Non-functional Semantics

- For ranking and selection
- Not constrained, any ontologies
- Example:

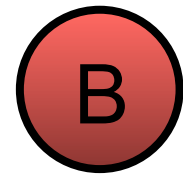
ex:PriceSpecification

`rdfs:subClassOf wl:NonFunctionalParameter .`



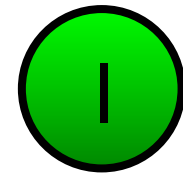
Behavioral Semantics

- For invocation, composition, process mediation
- Functionalities on operations
 - Capabilities, categories
- Client selects operation to invoke next
 - Instead of being strictly guided by an explicit process
- Example functional category for operations:
 - Web Architecture: interaction safety

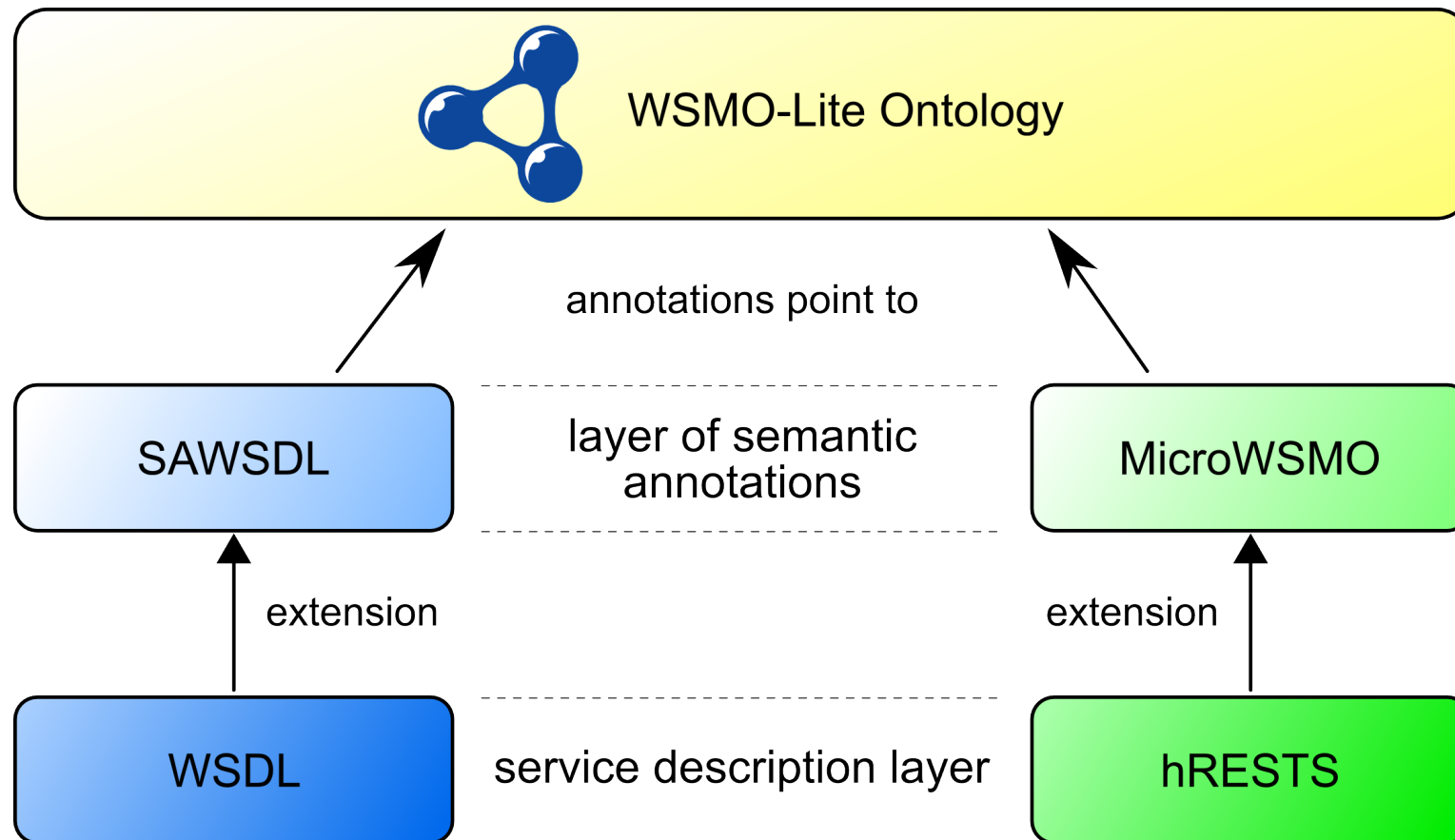


Information Semantics

- For invocation, composition, data mediation
- Not constrained, any ontologies
- Marked as `wl:Ontology`



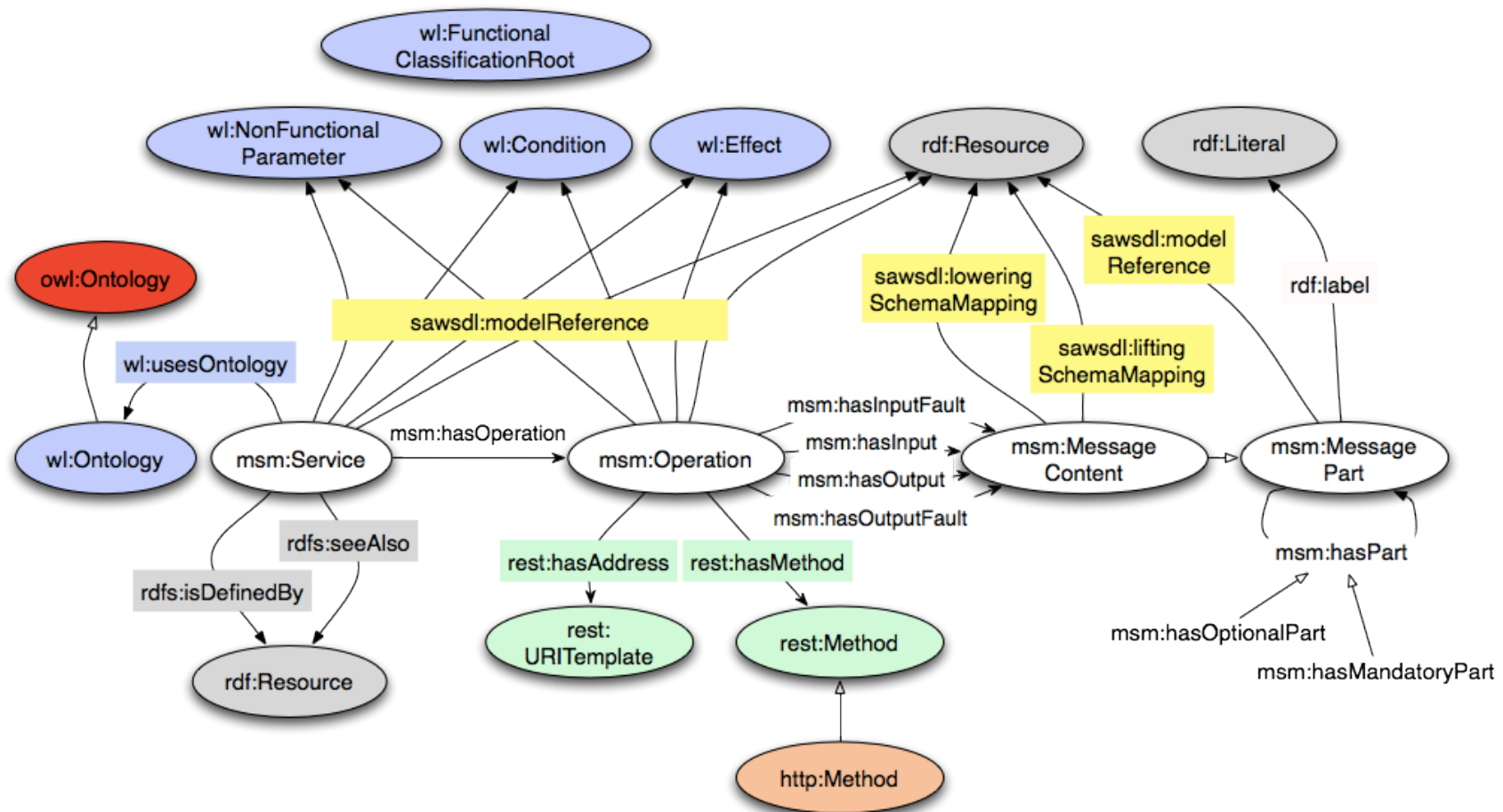
Semantics for Web Services and Web APIs



WSMO-Lite

- Defines a set of **common annotation** classes:
 - **FunctionalClassificationRoot**: instances are the roots of functional taxonomies
 - **NonFunctionalProperty**: subclasses are types of non-functional characteristics, e.g., QoS
 - **Condition**: logical condition that should hold before using a service/operation
 - **Effect**: logical condition that should hold after using a service/operation
- W3C member submission in July 2010

Conceptual Model



hRESTS

- "There's usually an HTML page"
- Identifying machine-readable parts
 - Service, its operations
 - Resource address, HTTP method
 - Input/output data format
- hRESTS microformat
 - Technically, a poshformat

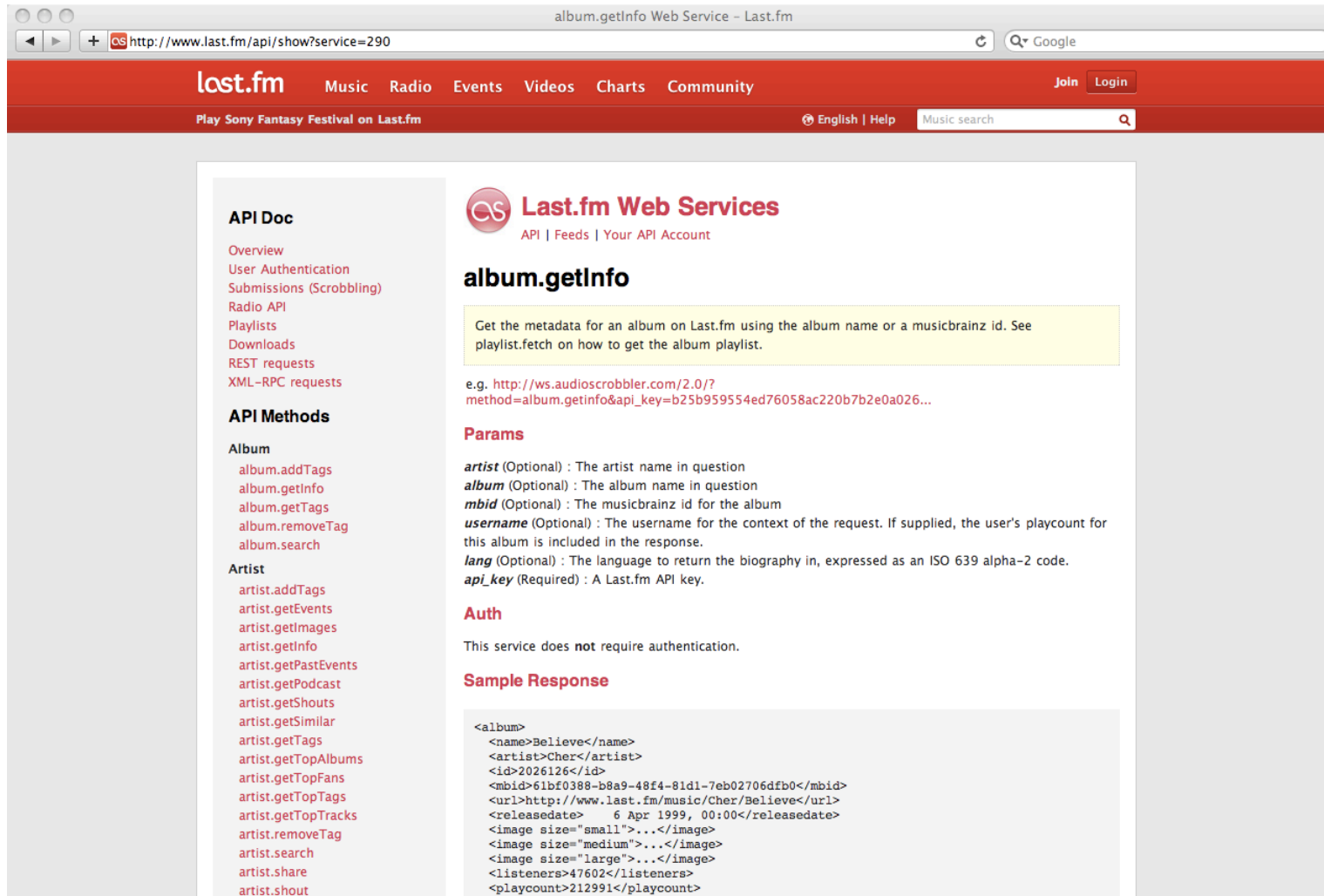
hRESTS

- HTML for RESTful Service Description
- Introduces the service model structure
 - service (+ label)
 - operations (+ address, method)
 - input, output
- Can also be in RDFa
- Basis for extensions:
 - MicroWSMO adds semantic annotations

MicroWSMO

- Extends hRESTS
 - model for model references
 - lifting, lowering
- Applies WSMO-Lite semantics presented earlier

Annotation Example



The screenshot shows a web browser window displaying the Last.fm Web Services API documentation. The browser's address bar shows the URL `http://www.last.fm/api/show?service=290`. The page has a red header with the Last.fm logo and navigation links for Music, Radio, Events, Videos, Charts, and Community. Below the header, there's a search bar and a "Music search" button. The main content area is divided into two columns. The left column, titled "API Doc", contains a list of API methods categorized under "Album" and "Artist". The right column, titled "Last.fm Web Services", provides details for the `album.getInfo` service. It includes a description, an example URL, a list of parameters, authentication requirements, and a sample XML response.

album.getInfo Web Service - Last.fm

CS **Last.fm Web Services**
API | Feeds | Your API Account

album.getInfo

Get the metadata for an album on Last.fm using the album name or a musicbrainz id. See playlist.fetch on how to get the album playlist.

e.g. `http://ws.audioscrobbler.com/2.0/?method=album.getInfo&api_key=b25b959554ed76058ac220b7b2e0a026...`

Params

artist (Optional) : The artist name in question
album (Optional) : The album name in question
mbid (Optional) : The musicbrainz id for the album
username (Optional) : The username for the context of the request. If supplied, the user's playcount for this album is included in the response.
lang (Optional) : The language to return the biography in, expressed as an ISO 639 alpha-2 code.
api_key (Required) : A Last.fm API key.

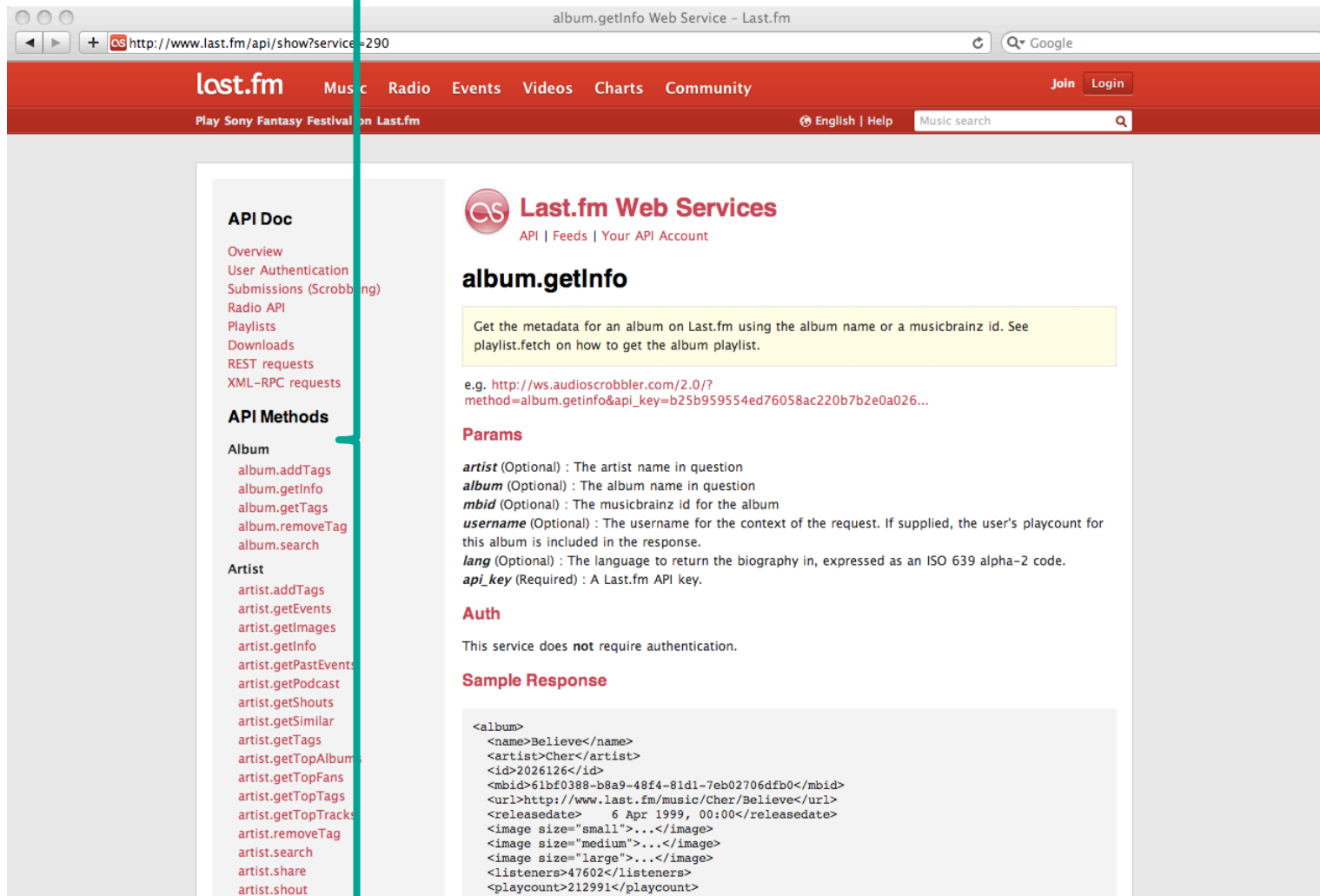
Auth

This service does **not** require authentication.

Sample Response

```
<album>
  <name>Believe</name>
  <artist>Cher</artist>
  <id>2026126</id>
  <mbid>61bf0388-b8a9-48f4-81d1-7eb02706dfb0</mbid>
  <url>http://www.last.fm/music/Cher/Believe</uri>
  <releasedate>6 Apr 1999, 00:00</releasedate>
  <image size="small">...</image>
  <image size="medium">...</image>
  <image size="large">...</image>
  <listeners>47602</listeners>
  <playcount>212991</playcount>
```


Annotation Example



album.getInfo Web Service - Last.fm

last.fm Music Radio Events Videos Charts Community Join Login

Play Sony Fantasy Festival on Last.fm English | Help Music search

API Doc

- Overview
- User Authentication
- Submissions (Scrobbling)
- Radio API
- Playlists
- Downloads
- REST requests
- XML-RPC requests

API Methods

Album

- album.addTags
- album.getInfo
- album.getTags
- album.removeTag
- album.search

Artist

- artist.addTags
- artist.getEvents
- artist.getImages
- artist.getInfo
- artist.getPastEvents
- artist.getPodcast
- artist.getShouts
- artist.getSimilar
- artist.getTags
- artist.getTopAlbums
- artist.getTopFans
- artist.getTopTags
- artist.getTopTracks
- artist.removeTag
- artist.search
- artist.share
- artist.shout

Last.fm Web Services

API | Feeds | Your API Account

album.getInfo

Get the metadata for an album on Last.fm using the album name or a musicbrainz id. See playlist.fetch on how to get the album playlist.

e.g. `http://ws.audioscrobbler.com/2.0/?method=album.getInfo&api_key=b25b959554ed76058ac220b7b2e0a026...`

Params

artist (Optional) : The artist name in question
album (Optional) : The album name in question
mbid (Optional) : The musicbrainz id for the album
username (Optional) : The username for the context of the request. If supplied, the user's playcount for this album is included in the response.
lang (Optional) : The language to return the biography in, expressed as an ISO 639 alpha-2 code.
api_key (Required) : A Last.fm API key.

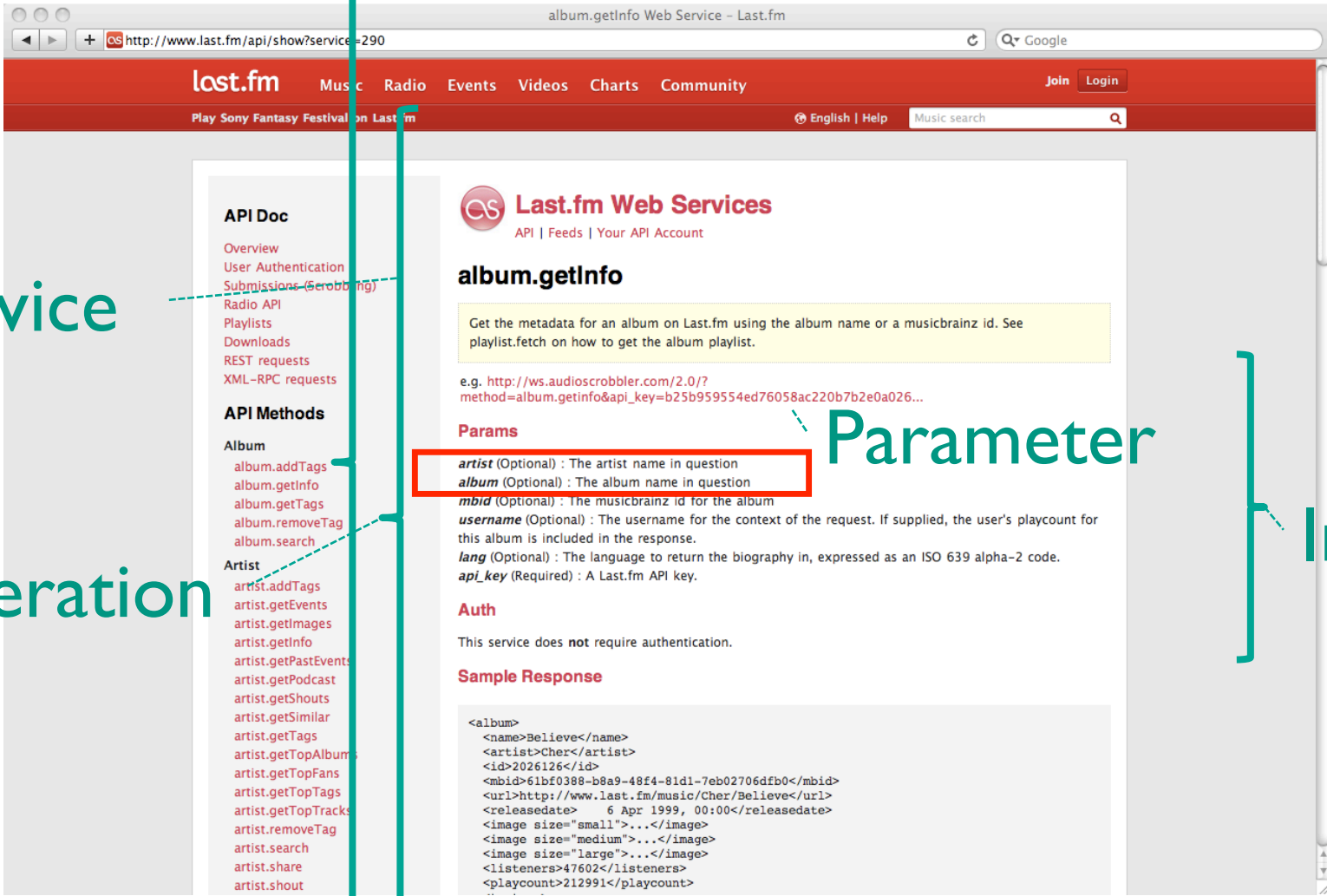
Auth

This service does **not** require authentication.

Sample Response

```
<album>
  <name>Believe</name>
  <artist>Cher</artist>
  <id>2026126</id>
  <mbid>61bf0388-b8a9-48f4-81d1-7eb02706dfb0</mbid>
  <url>http://www.last.fm/music/Cher/Believe</uri>
  <releasedate>6 Apr 1999, 00:00</releasedate>
  <image size="small">...</image>
  <image size="medium">...</image>
  <image size="large">...</image>
  <listeners>47602</listeners>
  <playcount>212991</playcount>
```

Annotation Example



Service

Operation

Parameter

Input

API Doc

- Overview
- User Authentication
- Submissions (Scrobbling)
- Radio API
- Playlists
- Downloads
- REST requests
- XML-RPC requests

API Methods

Album

- album.addTags
- album.getInfo
- album.getTags
- album.removeTag
- album.search

Artist

- artist.addTags
- artist.getEvents
- artist.getImages
- artist.getInfo
- artist.getPastEvents
- artist.getPodcast
- artist.getShouts
- artist.getSimilar
- artist.getTags
- artist.getTopAlbums
- artist.getTopFans
- artist.getTopTags
- artist.getTopTracks
- artist.removeTag
- artist.search
- artist.share
- artist.shout

Last.fm Web Services

API | Feeds | Your API Account

album.getInfo

Get the metadata for an album on Last.fm using the album name or a musicbrainz id. See playlist.fetch on how to get the album playlist.

e.g. `http://ws.audioscrobbler.com/2.0/?method=album.getInfo&api_key=b25b959554ed76058ac220b7b2e0a026...`

Params

- artist** (Optional) : The artist name in question
- album** (Optional) : The album name in question
- mbid** (Optional) : The musicbrainz id for the album
- username** (Optional) : The username for the context of the request. If supplied, the user's playlist for this album is included in the response.
- lang** (Optional) : The language to return the biography in, expressed as an ISO 639 alpha-2 code.
- api_key** (Required) : A Last.fm API key.

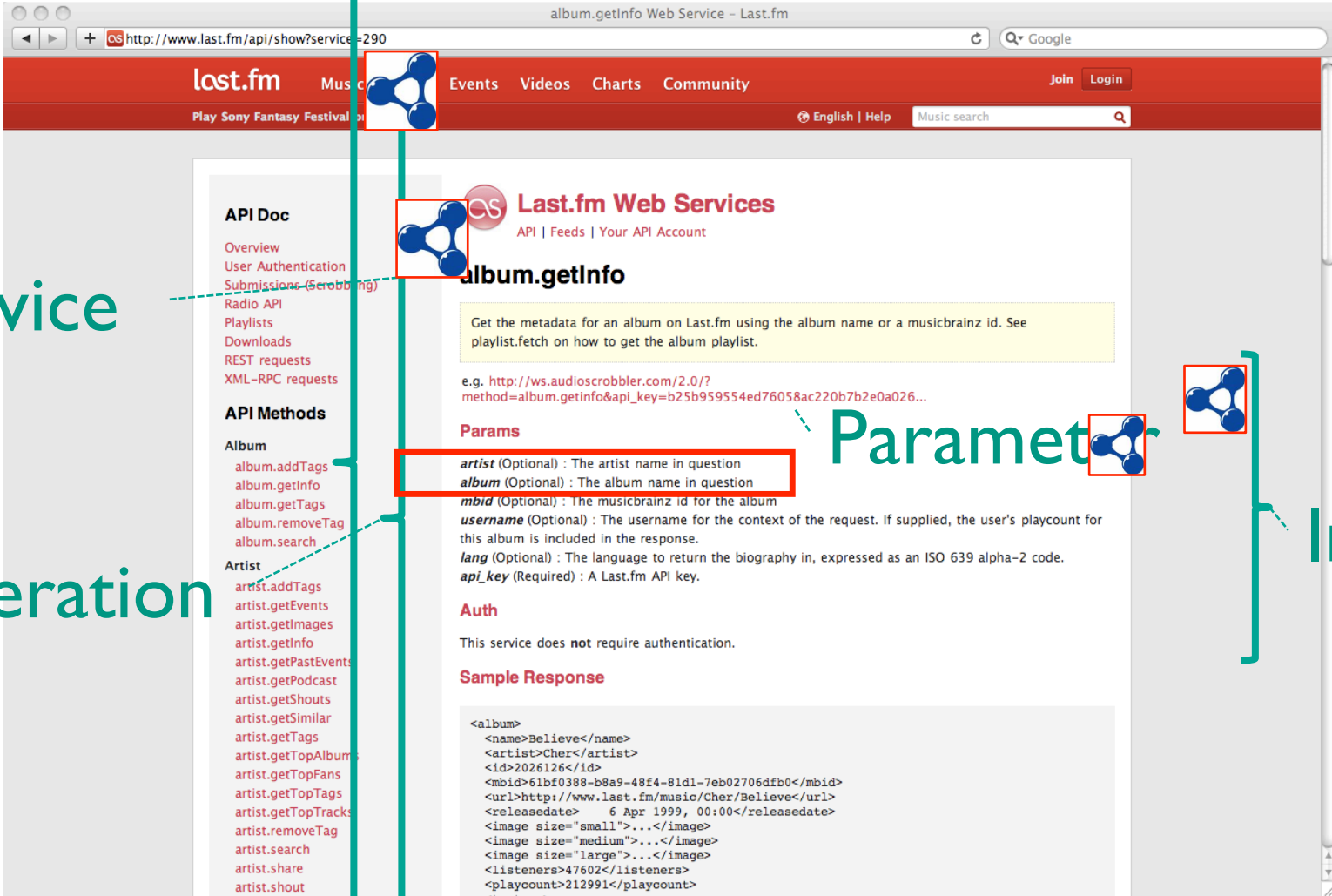
Auth

This service does **not** require authentication.

Sample Response

```
<album>
  <name>Believe</name>
  <artist>Cher</artist>
  <id>2026126</id>
  <mbid>61bf0388-b8a9-48f4-81d1-7eb02706dfb0</mbid>
  <url>http://www.last.fm/music/Cher/Believe</url>
  <releasedate> 6 Apr 1999, 00:00</releasedate>
  <image size="small">...</image>
  <image size="medium">...</image>
  <image size="large">...</image>
  <listeners>47602</listeners>
  <playcount>212991</playcount>
</album>
```

Annotation Example



Service

Operation

Parameter

Input

API Doc

- Overview
- User Authentication
- Submissions (Scrobbling)
- Radio API
- Playlists
- Downloads
- REST requests
- XML-RPC requests

API Methods

Album

- album.addTags
- album.getInfo
- album.getTags
- album.removeTag
- album.search

Artist

- artist.addTags
- artist.getEvents
- artist.getImages
- artist.getInfo
- artist.getPastEvents
- artist.getPodcast
- artist.getShouts
- artist.getSimilar
- artist.getTags
- artist.getTopAlbums
- artist.getTopFans
- artist.getTopTags
- artist.getTopTracks
- artist.removeTag
- artist.search
- artist.share
- artist.shout

Last.fm Web Services

API | Feeds | Your API Account

album.getInfo

Get the metadata for an album on Last.fm using the album name or a musicbrainz id. See playlist.fetch on how to get the album playlist.

e.g. `http://ws.audioscrobbler.com/2.0/?method=album.getInfo&api_key=b25b959554ed76058ac220b7b2e0a026...`

Params

- artist** (Optional) : The artist name in question
- album** (Optional) : The album name in question
- mbid** (Optional) : The musicbrainz id for the album
- username** (Optional) : The username for the context of the request. If supplied, the user's playlist for this album is included in the response.
- lang** (Optional) : The language to return the biography in, expressed as an ISO 639 alpha-2 code.
- api_key** (Required) : A Last.fm API key.

Auth

This service does **not** require authentication.

Sample Response

```
<album>
  <name>Believe</name>
  <artist>Cher</artist>
  <id>2026126</id>
  <mbid>61bf0388-b8a9-48f4-81d1-7eb02706dfb0</mbid>
  <url>http://www.last.fm/music/Cher/Believe</url>
  <releasedate> 6 Apr 1999, 00:00</releasedate>
  <image size="small">...</image>
  <image size="medium">...</image>
  <image size="large">...</image>
  <listeners>47602</listeners>
  <playcount>212991</playcount>
</album>
```

Annotation Example

```
<div class="service" id="service1">  
<h1 class="header">  
<span class="label" id="label2">Last.fm Web  
Services</span>
```

[...]

```
<div class="operation" id="operation1"><h1><span class="label"  
id="label1">artist.getInfo</span></h1>
```

```
<div class="wsdescription">Get the metadata for an artist on  
Last.fm. Includes biography.</div>
```

[...]

```
<div class="input" id="input1"><span class="param">artist</span>
```

[...]

Annotation Example

```
<div class="service" id="service1">
```

```
<h1 class="header">
```

```
<span class="label" id="label2">Last.fm Web  
Services</span>
```

```
<a rel="model" href="http://www.service-finder.eu/ontologies/  
ServiceCategories#Music"></a>
```

```
<div class="operation" id="operation1"><h1><span class="label"  
id="label1">artist.getInfo</span></h1>
```

```
<div class="wsdescription">Get the metadata for an artist on  
Last.fm. Includes biography.</div>
```

```
[...]
```

Resulting RDF Model

```
<rdf:Description rdf:about="http://iserve...">  
  <rdf:type rdf:resource="msm:Service"/>  
  <sawSDL:modelReference rdf:resource="http://www.service-finder.eu/  
ontologies/ServiceCategories#Music"/>  
</rdf:Description>
```

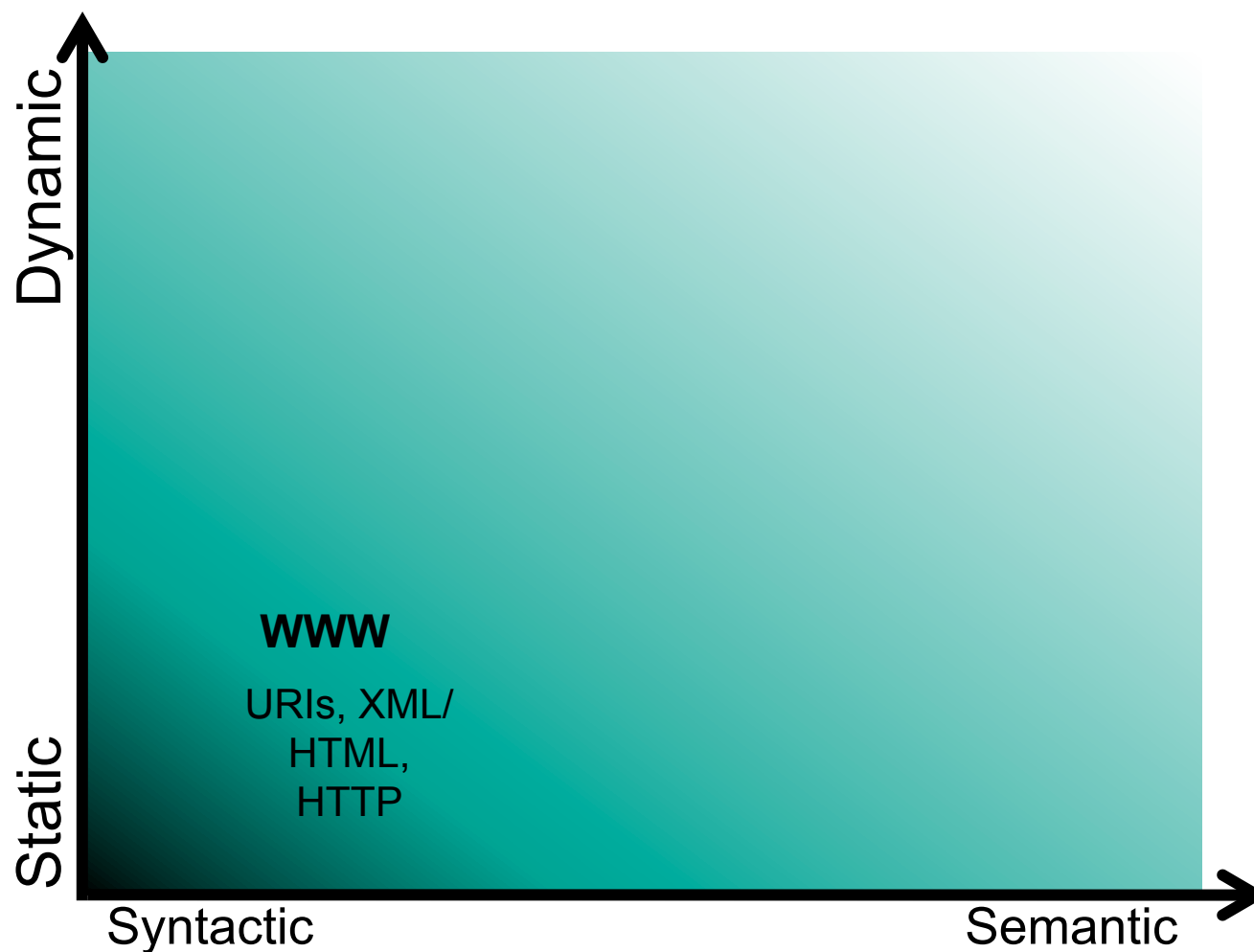
```
<rdf:Description rdf:about="http://iserve...">  
  <rdf:type rdf:resource="msm:Operation"/>  
</rdf:Description>
```

```
<rdf:Description rdf:about="http://iserve...">  
  <rdf:type rdf:resource="msm#MessageContent"/>  
</rdf:Description>
```

[...]

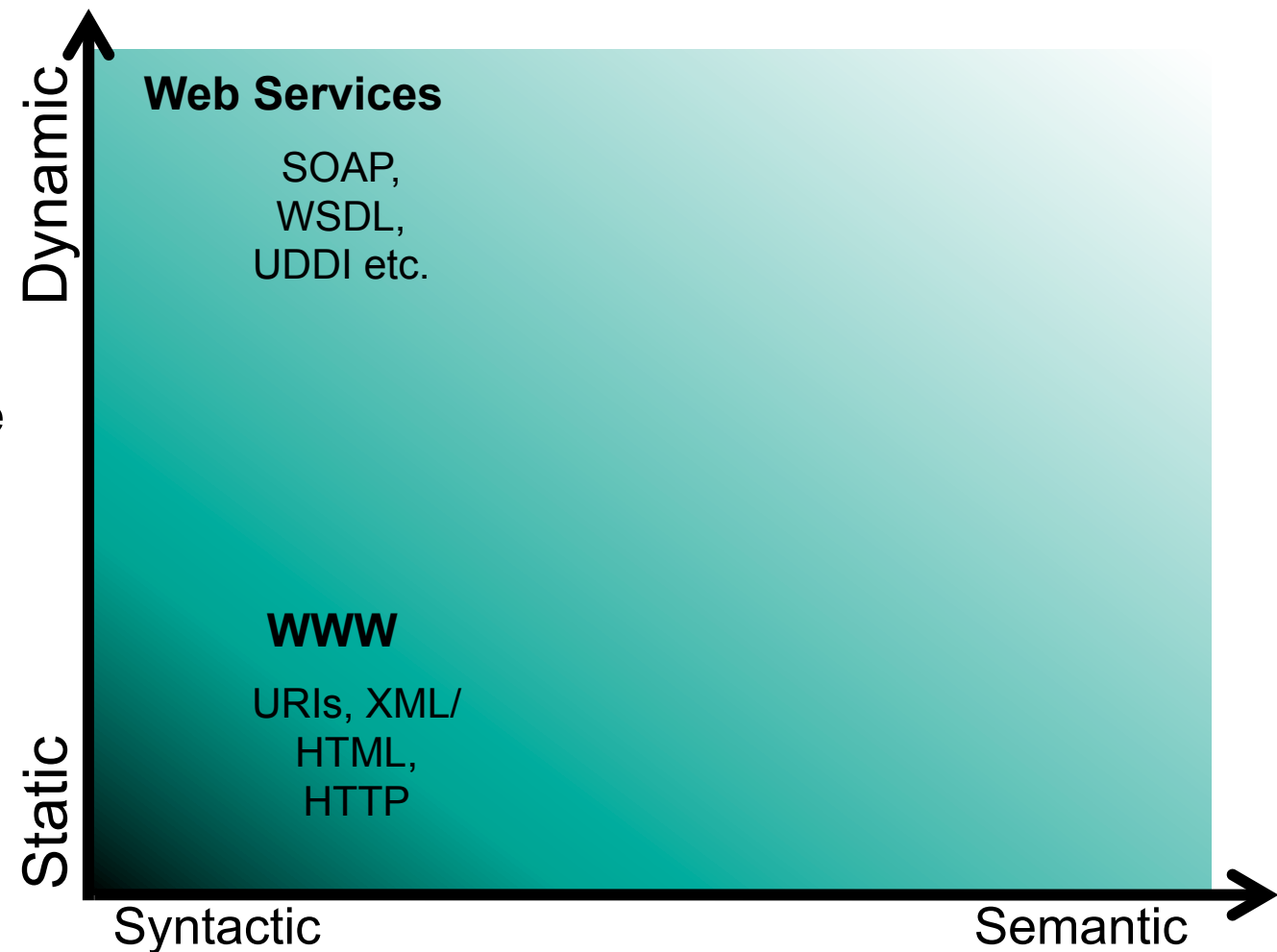
Semantic Web Services

- Just as Web-related technologies are applied....



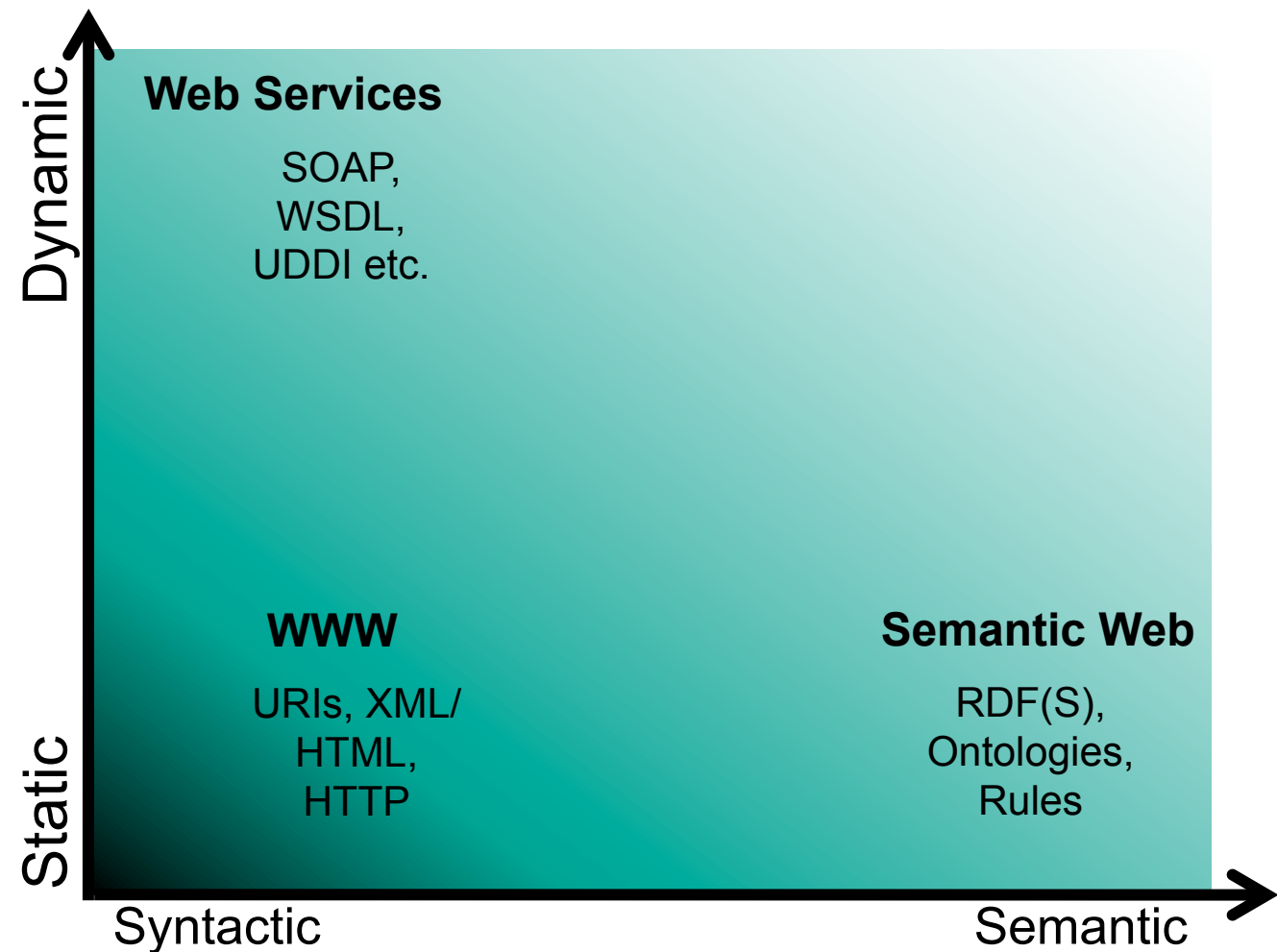
Semantic Web Services

- Just as Web-related technologies are applied as the basis of service descriptions to achieve dynamicity online



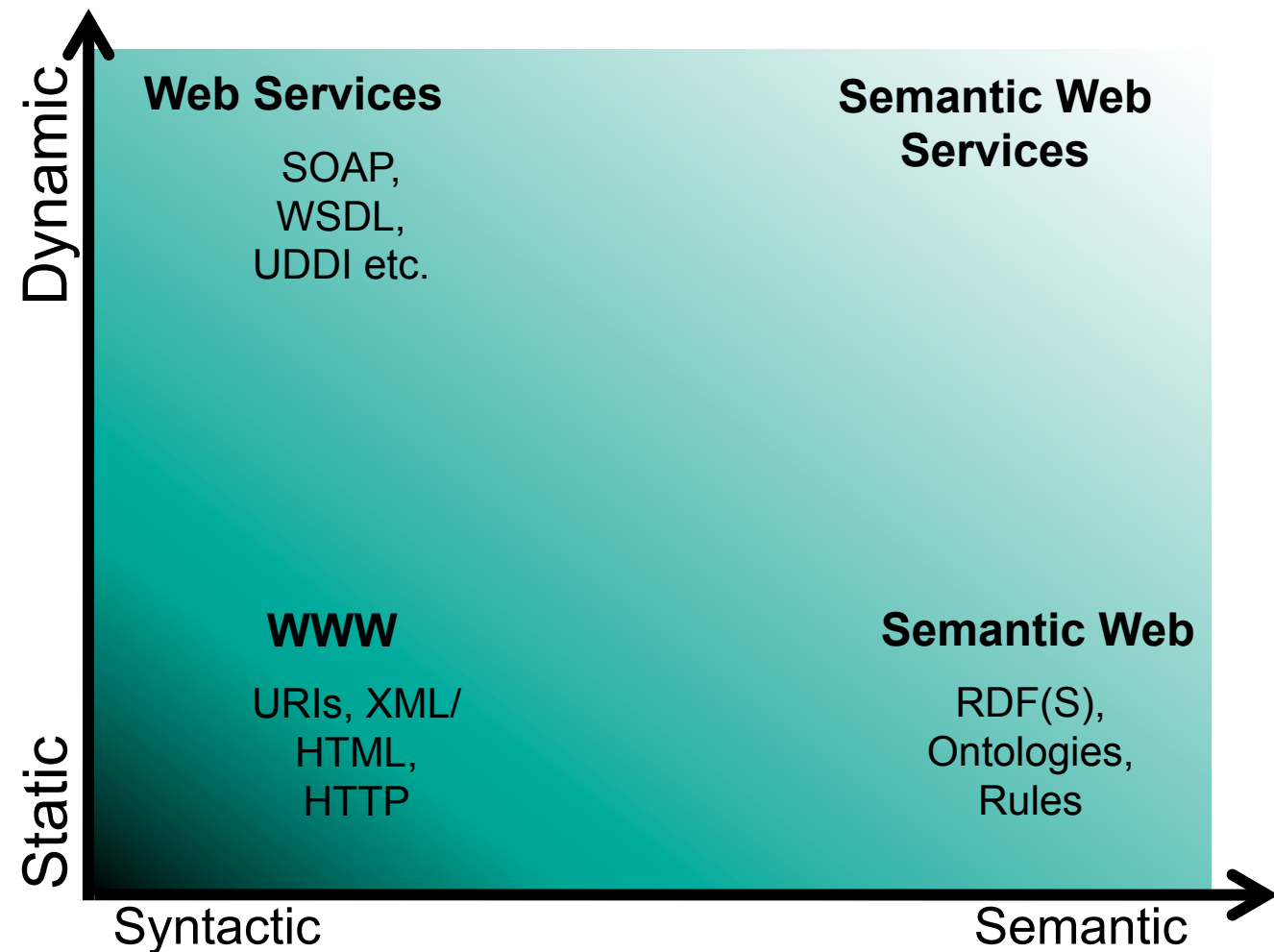
Semantic Web Services

- So the technologies that are applied to achieve explicit semantics on the Web...



Semantic Web Services

- So the technologies that are applied to achieve explicit semantics on the Web are combined with these to form SWS



Semantic Web Services Overview

- Semantic Web Services (SWS) are distinguished from the foregoing annotation-based approach by:
 - Building a top-down ontology model for services (rather than deriving this from bottom-up annotations)
 - Defining a large set of semantic extensions to the service model (cf. small core of annotation properties in WSMO-Lite)
 - Building Semantic Execution Environments for brokerage (repository maintenance, discovery, ranking, etc.) and execution of services
- Two approaches were documented as W3C member submissions:
 - OWL-S, submitted in November 2004
 - WSMO, submitted in June 2005

Bibliography & References

■ hRESTS and MicroWSMO

- <http://sweet.kmi.open.ac.uk/pub/microWSMO.pdf>
- 'Adapting SAWSDL for Semantic Annotations of RESTful Services', Maleshkova, M., Kopecky, J. and Pedrinaci, C., Beyond SAWSDL at OnTheMove Federated Conferences & Workshops, 2009

■ SAWSDL & WSMO-Lite

- <http://www.w3.org/TR/sawSDL/>
- 'WSMO-Lite Annotations for Web Services' Tomas Vitvar, Jacek Kopecky, Jana Viskova and Dieter Fensel, Proceedings of 5th European Semantic Web Conference (ESWC 2008)

Institut AIFB, KIT

Bitte fügen Sie hier ein Bild der Organisationseinheit auf der Masterfolie ein.